An Open Vehicle Routing Problem metaheuristic for examining wide solution neighborhoods

Emmanouil E. Zachariadis, Chris T. Kiranoudis

Department of Process Analysis and Plant Design, National Technical University of Athens, Athens, Greece, {ezach@mail.ntua.gr, kyr@chemeng.ntua.gr}

Abstract

This paper examines a practical transportation model known as the Open Vehicle Routing Problem (OVRP). OVRP aims at designing the minimum cost set of routes originating from a central depot for satisfying customer demand. Vehicles do not need to return to the depot after completing their delivery services. In methodological terms, we propose an innovative local search metaheuristic which examines wide solution neighborhoods. To explore these wide neighborhoods within reasonable computational effort, local search moves are statically encoded into Static Move Descriptor (SMD) entities. When a local search operator is applied to the candidate solution, only a limited solution part is modified. Therefore, to explore the next neighborhood only the tentative moves that refer to this affected solution part have to be re-evaluated, or in other words, only a subset of the SMD instances has to be updated, according to the modified solution state. The conducted search is efficiently performed by storing the SMD entities in Fibonacci Heaps, which are special priority queue structures offering fast minimum retrievals, insertions and deletions. To diversify the search, we employ a tabu scheme and a penalization strategy, both compatible with the SMD design. The proposed metaheuristic was tested on well-known OVRP instances, for two objective configurations. The first one primarily aims at minimizing the number of routes and secondarily minimizing the routing cost, whereas the second one only aims at minimizing the cost of the generated route set. For both configurations, it managed to produce fine results improving several previously best-known solutions.

***Keywords*:** *Open Vehicle Routing*, *Metaheuristics*, *Tabu Search*, *Computational Complexity*

1. Introduction

The distribution of goods is an important operational process which lies at the heart of modern business activity, and constitutes a significant part of the overall running costs of a company. For this reason, great research interest has been focused on the development of distribution systems, and on the design of solution approaches for effectively managing real-life logistics operations.

The most central and widely studied transportation model is the Capacitated Vehicle Routing Problem (CVRP), which as Li et al. [1] suggest is easy to state and difficult to solve. In specific, the CVRP model consists of a customer population with deterministic demands, and a central depot which acts as the base of a homogeneous fleet of vehicles. The aim of the CVRP is to design a set of Hamiltonian cycles (vehicle routes) starting and terminating at the central depot, such that the demand of customers is totally satisfied, each customer is visited once by a single vehicle, the total demand of the customers assigned to a route does not exceed vehicle capacity, and the overall travel cost of the designed route set is minimized.

In cases where industries do not own a vehicle fleet, or their private fleet is inadequate for fully satisfying customer demand, distribution services (or at least a part of them) are either entrusted to external contractors, or assigned to a hired vehicle fleet. In these cases, vehicles are not required to return to the central depot after their deliveries have been satisfied. The above described distribution model is referred to as the Open Vehicle Routing Problem (OVRP). It can be seen as a slight variant of the standard CVRP model by simply ignoring the return trip of the vehicles to the central depot. Therefore, the goal of the OVRP is to design a set of Hamiltonian paths (open routes) for satisfying customer demand. In terms of the OVRP objective, most researchers assume that the cost for hiring an additional vehicle far surpasses any travel cost savings achieved by this additional route [2, 3]; therefore their primary target is to minimize the number of required vehicles and secondarily minimize the total distance traveled. Although the minimization of routes is the most widely considered primary OVRP objective, researchers have also solved the OVRP aiming at solely minimizing the total distance traveled without taking into account the required fleet size.

In graph theoretic terms, OVRP is defined on a graph $G = (V, A)$, where $V = \{v_0, v_1, \ldots, v_n\}$ is the vertex set and $A = \{(v_i, v_j): v_i, v_j \in V, i \neq j, j \neq 0\}$ is the arc set. Vertex v_0 represents the central depot where a fleet of vehicles is located, each of them with maximum carrying load equal to Q. The remaining n vertices of $V \setminus \{v_0\}$ represent the customer set. With each customer vertex is associated a non-negative known demand q_i, whereas with each arc $(v_i, v_j) \in A$ is associated a cost c_{ij} which corresponds to the cost (travel time, distance) for transiting from v_i to v_j. As with most previous OVRP approaches, we consider that the cost matrix is obtained by calculating the Euclidean distances between vertex pairs, so that $c_{ij} = c_{ji}$ $(0 < i, j \leq n, i \neq j)$. The primary goal of the problem is to design the set of Hamiltonian paths (open routes) so that: (*a*) the size of the path set is minimized (minimization of vehicles), whereas the secondary objective is to (*b*) minimize the total cost of the generated paths. The following constraints must be satisfied: (*c*) every path originates from the central depot v_0, (*d*) each customer vertex is assigned to a single path, and (*e*) the total demand of the customer set assigned to a single path does not exceed the maximum carrying load Q of the vehicles (capacity constraint). Note that if the minimization of the fleet size is not considered to be the primary aim of the OVRP, only objective (*b*) is taken into account.

Our interest in the OVRP is motivated both by its theoretical and practical importance. In theoretical terms, the OVRP is an NP-hard combinatorial optimization problem: solving the OVRP to optimality implies that the best Hamiltonian path is obtained for each cluster of customers assigned to a vehicle. Since finding the best Hamiltonian path for a customer set is NP-hard, so is the OVRP [4]. From the commercial perspective, numerous real-world distribution activities fit into the OVRP framework. One of them is the home delivery of packages and newspapers [5]. Some additional OVRP example applications are the generation of airplane delivery routes [6], as well as the rail freight service [7], and school bus planning [8].

The aim of this paper is to present an innovative OVRP metaheuristic solution approach. In specific, the proposed local search algorithm, instead of applying simple customer swaps and relocations, has the ability to perform moves that involve larger customer sequences. The evaluation of these wider solution neighborhoods would require impractical computational time if not efficiently designed. Towards this aim, we make

use of the Static Move Descriptor concept [9], so as to reduce the complexity of applying these rich local search operators. To diversify the search, we use a tabu strategy together with a simple penalization policy both of them compatible with the proposed local search move encoding. Our algorithm was tested on well-known OVRP benchmark instances derived from the literature. It managed to produce fine quality results improving several previously reported best solutions.

The remainder of the present paper is as follows: Section 2 provides a literature review on published OVRP solution methodologies, followed by Section 3 which describes in detail our algorithmic development. Section 4 presents the computational results obtained by the proposed metaheuristic method. Finally, Section 5 concludes the paper.

2. Literature Review

As mentioned in the introductory Section of the article, the OVRP is an NP-hard combinatorial optimization problem; therefore to deal with OVRP instances of practical size, researchers have focused their interest on the development of effective heuristic and metaheuristic solution approaches.

The article of Schrage [10] which classifies the features encountered in practical routing problems was the first to distinguish between closed trips traveled by private vehicles, and open trips assigned to common carrier vehicles. The first solution approach for the OVRP is due to Bodin et al. [6]. Their paper deals with a practical routing problem faced by the airplane fleet of FedEx. In specific, airplanes layover at the end of their delivery routes, to later perform their pick-up trips. These delivery routes can be seen as an application of the OVRP, in the sense that airplanes do not return to the depot. Their solution approach is a variant of the Clarke & Wright (1964) algorithm adapted to the examined problem. Sariklis and Powell [11] were the first to formally introduce the OVRP model. To solve the OVRP, they present a heuristic method based on a minimum spanning tree combined with a penalization procedure. Brandão [4] presents a tabu search procedure which makes use of customer insertion and swap local search operators. Tarantilis et al. [12, 13] have published two studies on the OVRP, both belonging to the threshold accepting category of algorithms. The former work [12] proposes an annealing-

based method that utilizes a backtracking policy of the threshold value when no acceptances of feasible solutions occur during the search process, whereas the latter study [13] presents a single-parameter metaheuristic method that exploits a list of threshold values to intelligently guide an advanced local search method. Tarantilis et al. [14] have also published an adaptive memory approach for the OVRP. Their approach involves a pool of routes that belong to the highest quality solutions encountered through the search process. Sequences of customers are extracted from the adaptive memory to form new partial solutions later to be improved by a tabu search procedure. The routes of these improved solutions are used to update the adaptive memory contents, forming in this way a cyclic algorithm. Note that the aforementioned three works aim at solely minimizing the total distance of the open routes, disregarding the required fleet size. Fu et al. [2, 3] propose a metaheuristic framework which constructs an initial OVRP solution via a farthest-first heuristic. This solution is then improved by a tabu search method which employs the well-known relocation, swap, and 2-opt operators. Li et al. [15] have dealt with the OVRP by developing a local search metaheuristic algorithm which uses the concept of record-to-record travel [16]. In their work, they introduce eight large-scale OVRP instances which have served as a comparison basis for the effectiveness of recent OVRP methodologies. These recent works include the general routing heuristic of Pisinger and Ropke [17] which has been effectively applied to the OVRP variant. Their approach involves the application of the adaptive large neighborhood search framework. Derigs and Reuter [18] have proposed a parameter-free method based on the attribute based hill climber concept [19]. The work of Repoussis et al. [20] proposes an evolutionary algorithm for the OVRP. The parent solution population creates an intermediate population of offspring solutions via mutation. These offspring solutions are improved by a hybridization of tabu search [21] and guided local search [22], to update the solution population. Fleszar et al. [23] propose a variable neighborhood search [24] approach which examines neighborhood structures defined by route segment reversals and exchanges. Finally, Li et al. [25] present a hybrid metaheuristic based on ant colony optimization and tabu search. The solution obtained by the aforementioned solution development is further improved via a post-optimization tabu search procedure. Except for the aforementioned heuristic and metaheuristic solution approaches, Letchford et al

[26] have proposed a branch-and-cut-algorithm for the OVRP. The authors provide an OVRP integer programming formulation together with some valid inequalities. Their algorithm is capable of solving to optimality small- to medium-sized OVRP instances. In addition, for the large scale instances, they provide lower bounds which are helpful for assessing the effectiveness of approximate solution methodologies.

3. The Proposed Algorithm

As mentioned in Section 1, the proposed OVRP metaheuristic is based on the Static Move Descriptor (SMD) concept [9], a strategy which reduces the computational complexity required for applying local search-based methods. Reducing the complexity for performing local search allows our algorithm to explore rich solution neighborhoods within manageable computational effort. In this section, we firstly provide a brief presentation of the SMD concept, followed by an in-depth description of the way in which the SMD strategy is adapted for the applied local search operators. Finally, the overall algorithmic framework is provided.

3.1. Introduction to the static move descriptor concept

A wide collection of the most effective metaheuristic strategies (Tabu Search, Variable Neighborhood Search, Guided Local Search etc.) fit in the category of local search-based methods. The basic principle of local search is to explore the search space by iteratively moving from a candidate solution to a new solution which belongs to the *neighborhood* of the former. The neighborhood of a solution is composed by every solution that can be generated from it by performing a systematic modification (*move*) on the solution structure. Thus, the computational effort for applying a local search scheme for dealing with an optimization problem is mainly determined by the complexity demanded for examining the solution neighborhoods (or, in other words, the set of tentative moves), a task which is repeatedly executed within a local search framework.

The central underlying idea of the SMD concept is that when moving from one solution to another, only a limited solution part is affected; therefore to examine the subsequent solution neighborhood, only the tentative moves that are associated with this modified solution part have to be re-evaluated. On the other hand, tentative moves that refer to

unmodified solution characteristics have already been evaluated (during previous neighborhood explorations), so if they are appropriately stored, their recalculation is unnecessary.

To realize this idea, we propose the Static Move Descriptors (SMD) entities which encode tentative local search moves in a static (solution-independent) way. Each SMD instance corresponds to a specific move, while it also includes the cost involved for performing this move. Therefore, within a local search procedure, each time a move is applied to a candidate solution, only the costs of the SMD entities that refer to the modified solution part have to be re-evaluated. To implement the best admissible local search scheme (moving to the highest-quality neighboring solution), SMD instances are stored in Fibonacci Heaps [27] which are special priority queue data structures with the following key capabilities: constant time minimum retrievals and insertions, and logarithmic time deletions.

3.2. The local search operators and their SMD representation

As previously mentioned, having realized a local-search complexity reduction strategy, we aimed at designing and applying powerful local search operators, the application of which would be computationally prohibitive without the complexity reduction scheme. In specific, instead of the classic 1-0 and 1-1 exchanges, we propose a Variable Length Bone Exchange (VLBE) local search operator, which exchanges the positions of any *bone* containing from 0 to μ customers, where bone denotes a customer sequence present in a candidate solution. Apart from the aforementioned VLBE operator, we also employ the classic 2-opt move which replaces any pair of solution arcs with a new arc pair. In the following, we provide analytic presentations for these two local search operators, and their SMD representation.

3.2.1. The Variable Length Bone Exchange operator

The VLBE operator exchanges the positions of any customer sequence pair containing from 0 to μ customers. Given that the number of customer pairs is $O(n^2)$, and that the number of combinations of the two bone lengths involved in the move is $O(\mu^2)$, the cardinality of the neighborhood structure defined by VLBE is $O(\mu^2 \cdot n^2)$. Therefore, the

complexity required for exhaustively exploring the neighborhoods defined by the VLBE operator is $O(\mu^2 \cdot n^2)$, which makes exhaustive neighborhood evaluation almost unmanageable for instances of practical size and significant μ values.

To encode the VLBE tentative moves into SMD instances, we use the following rationale: For every pair of non-identical vertices v_i and v_j, a collection of SMD instances is created, each of them corresponding to a particular VLBE move. Every such SMD instance contains the following information: The pair of nodes to which it belongs (denoted by n_1 and n_2), two bone lengths (denoted by *n1_len*, *n2_len*) both of them ranging from 0 to μ, and the cost involved for performing the encoded move. No SMD instance is created for *n1_len* = 0 and *n2_len* = 0, thus, for each pair of vertices, in total $(\mu + 1)^2 - 1$ SMD instances are generated. When a VLBE SMD with n_1 = A, n_2 = B, *n1_len* = *a*, and *n2_len* = *b* is applied to an OVRP solution, the following structural modification is performed: The bone beginning after A and containing *a* customers, and the bone beginning after B and containing *b* customers exchange their positions.

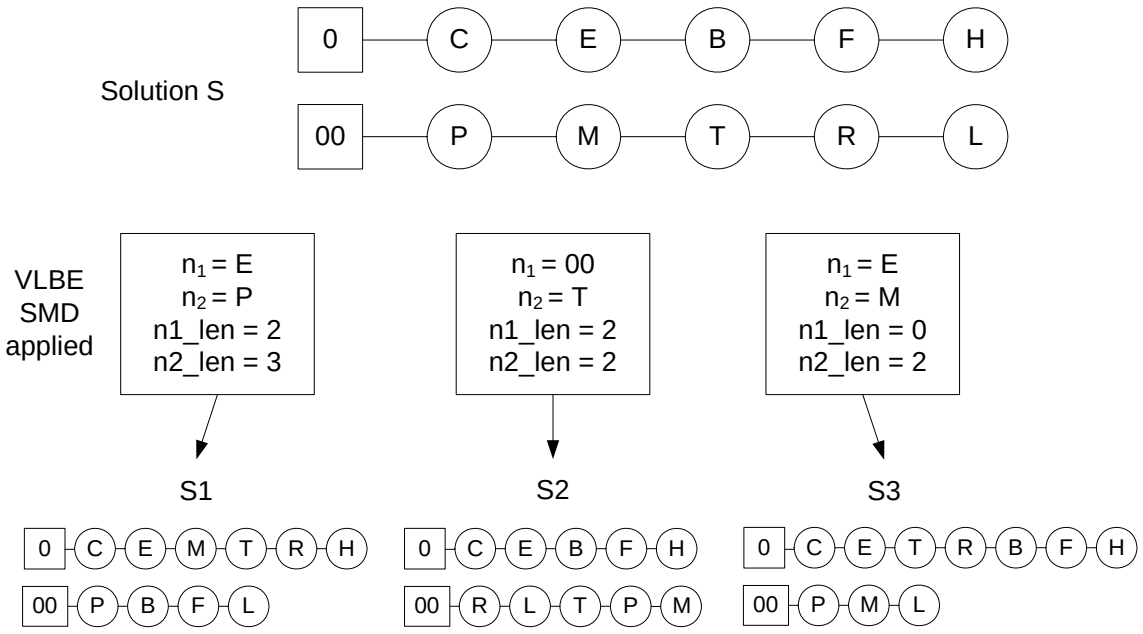

Fig 1. The VLBE SMD representation

To exhaustively describe the VLBE neighborhood in total $((n+K)!/(2!(n+K-2)!))\cdot((\mu+1)^2-1)$ SMD instances are required. The first factor corresponds to the 2-combinations without repetition of the n customers and K depot occurrences, where K is the total number of routes present in the solution, whereas the second factor corresponds to the total SMD instances per vertex pair. Three example VLBE SMD applications to an arbitrary OVRP solution of two routes and ten customers are illustrated in Fig.1. In specific, the first VLBE SMD applied to solution S has n_1 = E, n_2 = P, *n1_len* = 2, and *n2_len* = 3. The bone beginning after E and containing two nodes (B-F) is swapped with that beginning after P and containing three customers (M-T-R) to form solution S1. For the second illustrated intra-route SMD, we have n_1 = 00, n_2 = T, *n1_len* = 2, and *n2_len* = 2. The node 00 corresponds to the depot vertex of the second OVRP route. Therefore, the two-customer bone beginning after the 00 depot vertex (P-M) exchanges positions with bone R-L to form solution S2. The third example move consists of relocating a single bone, as *n1_len* = 0. In detail, the two-customer (*n2_len* = 2) bone beginning after n_2 = M (T-R) is relocated next to n_1 = E. Note that when either *n1_len* or *n2_len* is equal to zero, the SMD encodes a relocation move rather than an exchange one.

3.2.2. The 2-opt operator

The 2-opt local search operator replaces two arcs present in the solution. The mechanism of the move depends on the arc pair involved in the move. If both arcs belong to the same route (intra-route 2-opt move), then these arcs are deleted, two new arcs are generated, and the path lying between the deleted arcs is reversed. If the arcs to be removed belong to different routes (inter-route 2-opt move), then their deletion implies the division of these routes to their initial and final segments. The generated arc pair connects the initial segment of the first route to the terminating segment of the second one and vice versa.

To encode 2-opt moves using the SMD representation, we generate one 2-opt SMD instance per distinct vertex pair. Each 2-opt SMD instance contains the pair of nodes to which it belongs (denoted by n_1 and n_2), and the cost involved for implementing the encoded move. When a 2-opt SMD with n_1 = A, n_2 = B, is applied to an OVRP solution, the following structural change takes place: If vertices A and B belong to different routes,

the route segment beginning at the depot and terminating at A is connected to the route path beginning after B. Analogously, the route segment originating from the depot and terminating at B is connected to the route segment which begins after vertex A. Otherwise, if vertices A and B are assigned to the same route and B precedes A in the route vector, the A and B values are swapped. Then, vertex A is connected to B, by reversing the path beginning after A and terminating at B.

To exhaustively describe the 2-opt solution neighborhood, in total $((n + K)!/(2!(n + K - 2)!))$ SMD instances are required, corresponding to the 2-combinations without repetition of the n customers and K depot occurrences, where K is the total number of routes present in the solution. Three example 2-opt SMD moves to an arbitrary OVRP solution of two routes and ten customers are illustrated in Fig.2. The first illustrated SMD has n_1 = E, and n_2 = M. As previously mentioned, the initial segment of the first route (0-C-E) is connected to the terminating segment of the second one (T-R-L) and the initial segment of the second route (00-P-M) is connected to the terminating segment of the first one (B-F-H), so that solution S1 is generated. In an analogous way, the first route of S2 is formed by linking the initial and terminating segments of the first and the second route, respectively (0-C and P-M-T-R-L), whereas the second S2 route is obtained by connecting the initial and terminating segments of the second and the first route, respectively (00 and E-B-F-H). The third SMD instance represents an intra-route move. Customer C is connected to customer F by replacing arcs CE and FH with CF and EH. In addition, as seen in Figure 2, the path originating after node C and terminating at F is reversed to form solution S3.

At this point, we should note that for problem instances involving few customers per route (relatively close to the considered bone length μ), there are several pairs of 2-opt and VLBE instances which encode the same local search move. In specific, for problems with up to μ customers assigned to each vehicle, 2-opt SMD instances should be designed for representing only the intra-route 2-opt moves, as the inter-route part of the 2-opt neighborhood is fully described by the VLBE operator. For our research, however, which involved solving problem instances with large customer sequences per route, considering the complete 2-opt neighborhood structure was proven useful, as it enabled the exchanges of bones which contained more than μ customer vertices.

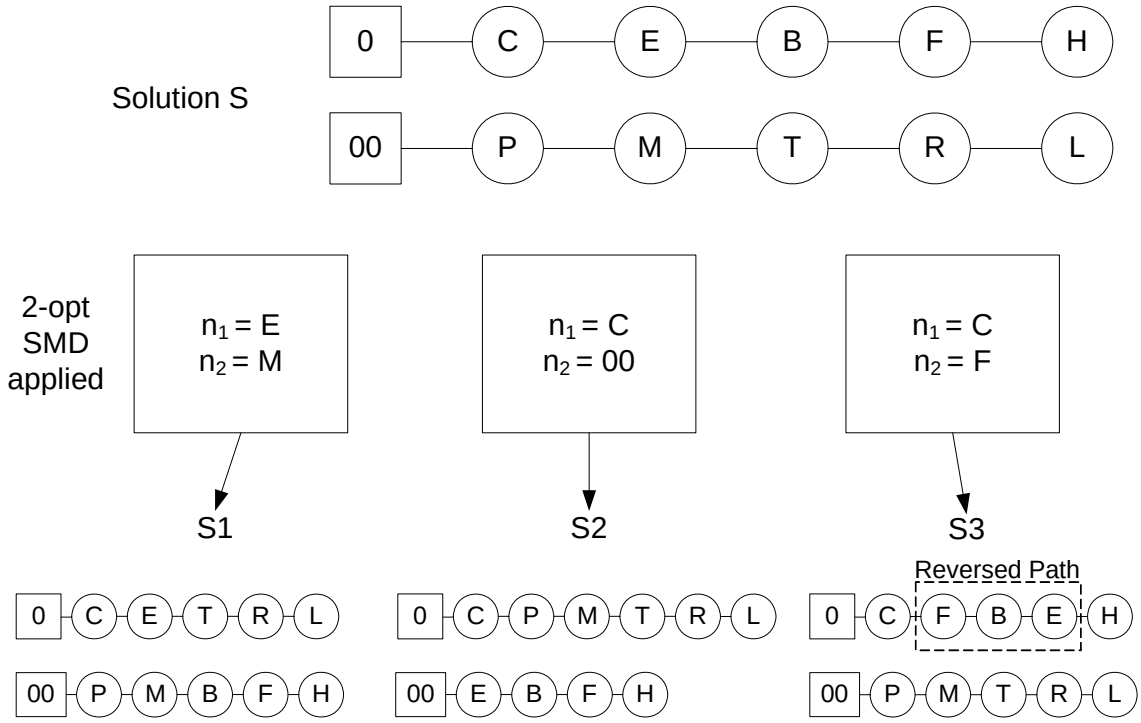

Fig 2. The 2-opt SMD representation

3.3. Feasibility checking for the 2-opt and VLBE local search operators

As will be later shown, the proposed algorithmic framework does not allow infeasible tunneling, or in other words, it only applies SMD instances which do not lead to capacity constraint violations. To examine feasibility of the VLBE and 2-opt SMD instances in constant time, we use the following rationale: Let $d_{\mathrm{RT}}(i)$ be the total product demand of the route RT bone beginning at the depot and containing i customers, with i varying from 0 to z_{RT}, where z_{RT} denotes the total number of customers assigned to route RT. Obviously, $d_{RT}(0)$ is equal to zero (depot product demand), and $d_{\mathrm{RT}}(z_{RT})$ is equal to the total product demand of the customer set assigned to route *RT*. In addition, let $pos(v_i)$ denote the position of vertex v_i in its route.

An inter-route SMD instance with n_1 = A, n_2 = B, $n1\_len = a$, and $n2\_len = b$, with A and B assigned in routes RTA and RTB, respectively, is feasible if the following two conditions hold:

- $d_{\mathrm{RTA}}(z_{\mathrm{RTA}}) - (d_{\mathrm{RTA}}(pos(\mathrm{A}) + a) - d_{\mathrm{RTA}}(pos(\mathrm{A}))) + (d_{\mathrm{RTB}}(pos(\mathrm{B}) + b) - d_{\mathrm{RTB}}(pos(\mathrm{B}))) \leq Q$,

- $d_{RTB}(z_{RTB})$ - ($d_{RTB}(pos(B) + b)$ - $d_{RTB}(pos(B))$) + ($d_{RTA}(pos(A) + a)$ - $d_{RTA}(pos(A))$) ≤ Q.

An inter-route 2-opt SMD instance with n_1 = A, n_2 = B, with A and B assigned in routes RTA and RTB, respectively, is feasible if both of the following conditions hold:

- $d_{RTA}(pos(A)) + (d_{RTB}(z_{RTB}) - d_{RTB}(pos(B))) \leq Q$,
- $d_{RTB}(pos(B)) + (d_{RTA}(z_{RTA}) - d_{RTA}(pos(A))) \leq Q$.

Note that all VLBE and 2-opt SMD instances encoding intra-route moves are feasible because they do not cause any effect on the total product quantities assigned to the vehicles.

3.4. Keeping the SMD cost tags updated

As previously mentioned, when the operators described in 3.2 are performed on a candidate solution, the cost tags of the SMD subset that refers to the modified solution part have to be updated. In the following, we provide the rules of the necessary cost updates, together with an analysis on their total number.

3.4.1. Cost update rules for the application of a VLBE move

Consider that a VLBE SMD instance with n_1 = A, n_2 = B, len_n1 = a, and len_n2 = b is applied to a candidate OVRP solution, as in Fig. 3. Let *pred*(A) and *pred*(B) denote the bones terminating before nodes A, and B respectively and containing μ customers. In addition, let *exch*(A) and *exch*(B) denote the two bones exchanged (the final nodes of the exchanged bones are excluded) which include up to μ-1 customers, whereas *lst*(A) and *lst*(B) denote the ending nodes of the exchanged bones.

To keep every VLBE SMD cost updated, the cost tag of every VLBE SMD instance with either its n_1 or n_2 equal to A, B, *lst*(A), and *lst*(B) must be calculated. In addition, the VLBE SMD instances whose n_1 or n_2 belongs to *pred*(A) (or *pred*(B)) and the relevant bone length referring to the route segment after node A (node B) must be re-evaluated according to the modified solution state. Finally, every VLBE SMD instance with n_1 or n_2 belonging to *exch*(A) or *exch*(B) and the relevant bone length reaching after the exchanged bones must also be updated. Obviously, at most $O(\mu^3 \cdot n)$ updates are required,

as there are $O(\mu \cdot n)$ groups of VLBE SMD instances with their n_1 or n_2 belonging to {A, B, *lst*(A), *lst*(B), *pred*(A), *pred*(B), *exch*(A), *exch*(B)} and each of these groups contains $O(\mu^2)$ (2-combinations of bone lengths) VLBE SMD instances.

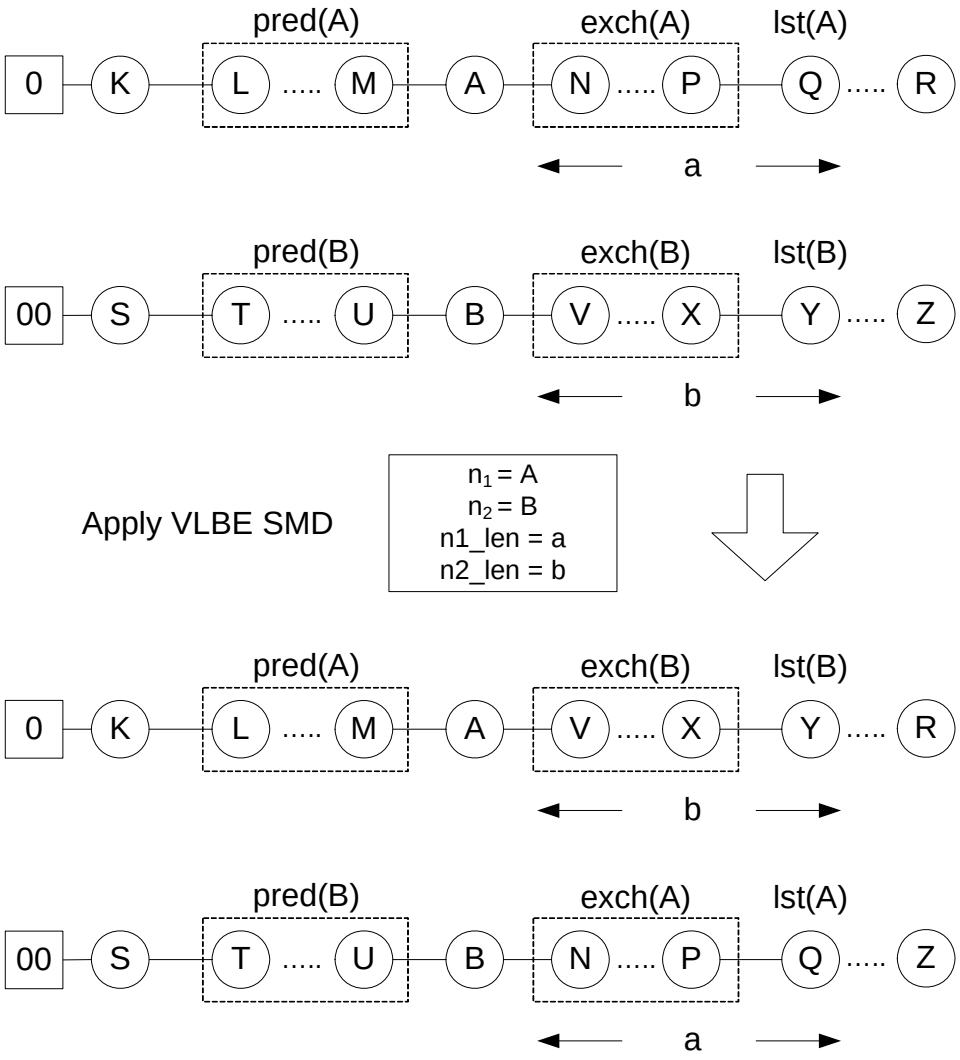

Fig 3. Applying a VBLE SMD to an OVRP solution

Regarding the cost of the 2-opt tentative moves, firstly the cost of every 2-opt SMD instance with either its n_1 or n_2 value equal to A, B, *lst*(A), and *lst*(B) must be updated. Apart from these $O(n)$ updates, when the applied VLBE SMD encodes an inter-route move, some additional 2-opt SMD cost tags need to be updated to capture the fact that the vertices of the exchanged bones are moved from their current route to another one. In

specific, let *init*(v) and *fin*(v) denote the route segments lying before and after node v, respectively (node v is excluded in both cases). Then for the move presented in Fig 3, the following 2-opt SMD instances have to be updated: those with their one node value (n_1 or n_2) equal to *exch*(A) or *exch*(B), and their other node value belonging to {*init*(A), *init*(B), *fin*(*lst*(A)), *fin*(*lst*(B))}. The necessary updates in this case are bounded by $O(\mu \cdot n)$, as at most $O(\mu)$ nodes are included in the exchanged bones, and up to $O(n)$ nodes are contained in the initial and terminating segments of the routes involved in the move.

3.4.2. Cost update rules for the application of a 2-opt move

The mechanism for updating the cost tags of the VLBE and 2-opt SMD instances when a 2-opt move is applied depends on whether this move involves a single route or a route pair:

Inter-Route 2-opt move:

Consider that an inter-route 2-opt SMD with n_1 = A and n_2 = B is applied to an OVRP solution, and let *pred*(A) and *pred*(B) denote the bones terminating before nodes A, and B, respectively and containing μ customers, as illustrated in Fig. 4.

In terms of the necessary VLBE cost updates, every SMD with n_1 or n_2 belonging to *pred*(A) or *pred*(B) and relevant bone lengths that point after A and B, respectively, has to be updated. Thus, $O(\mu^3 \cdot n)$ updates are required in total. Additionally, the cost tag of every VLBE SMD with n_1 or n_2 equal to A or B has to be re-calculated, corresponding to $O(\mu^2 \cdot n)$ updates.

Regarding the necessary cost updates for the 2-opt operator, $O(n)$ cost re-evaluations must be made corresponding to the SMD instances with node values equal to A or B. In addition, we have to capture the fact that the initial segment of the first route is connected to the final segment of the second one, and vice versa. To do so, let *init*(v) and *fin*(v) denote the route segments lying before and after node v, respectively. The 2-opt SMD instances with their one node value (n_1 or n_2) within *init*(A), and their other node value belonging either to *fin*(A) or *fin*(B) have to be updated. In an analogous way, every 2-opt SMD with one node value included in *init*(B), and the other node value within either *fin*(A) or *fin*(B) must be re-evaluated. The amount of required cost evaluations depends

on the number of vertices within the routes involved in the move. If *n_rt* denotes the total number of customers assigned to the route pair involved in the move, at most $O(n\_rt^2)$ updates are required, which for instances involving few customer per route, as well as, for significant μ values, is considerably lower compared to the cost updates required when a VLBE move is applied ($O(\mu^3 \cdot n)$).

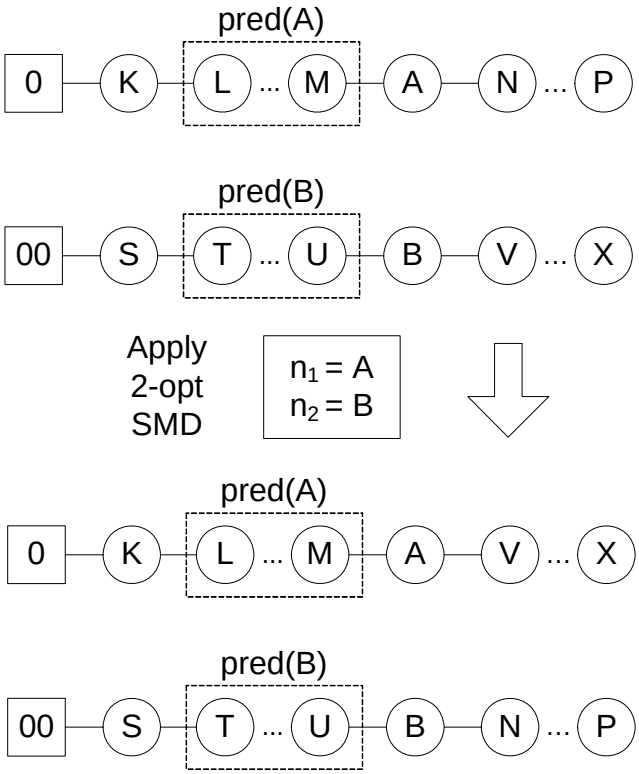

Fig 4. Applying an inter-route 2-opt SMD to an OVRP solution

Intra-Route 2-opt move:

Consider that an intra-route 2-opt SMD with n_1 = A and n_2 = B is applied to an OVRP solution, as illustrated in Fig.5. Let *rev*(A) denote the route path which begins after node A, includes *n_rev* customers and is reversed when the intra-route 2-opt is applied, and *pred*(A) be the route path before A containing up to μ customers.

In terms of the VLBE operator, the VLBE SMD instances with their node values contained in *pred*(A) and the relevant bone length pointing after A are updated. Thus, $O(\mu^3 \cdot n)$ cost re-evaluations are required. Furthermore, the cost of the VLBE SMD

instances for which n_1 or n_2 is equal to A and the relevant bone length does not exceed *n_rev* must be re-calculated regarding the modified solution, corresponding to $O(\mu^2 \cdot n)$ SMD cost updates. Finally, every VLBE instance with one node value belonging to *rev*(A) (for all 2-combinations of bone lengths) must also be updated. The number of required updates is $O(n \cdot n\_rev \cdot \mu^2)$.

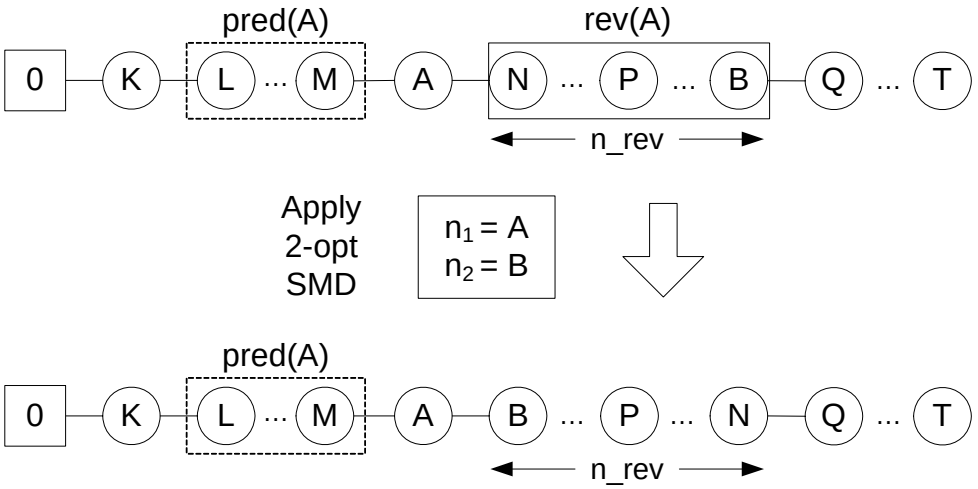

Fig 5. Applying an intra-route 2-opt SMD to an OVRP solution

Regarding the 2-opt neighborhood, in total $O(n \cdot n\_rev)$ updates are necessary, corresponding to the 2-opt SMD instances with one node value either equal to A or contained in *rev*(A).

Note that the updates required for both the VLBE ($O(\mu^2 \cdot n \cdot n\_rev)$), and 2-opt ($O(n \cdot n\_rev)$) neighborhood structures depend on the size of the route path reversed, and therefore on the characteristics of the OVRP instance solved. The aforementioned complexity levels are comparable to those required for the cost updates of the VLBE moves, in cases of significant μ values and limited sizes of the reversed route segments.

3.5. The overall structure of the proposed solution approach

Our algorithmic framework is initiated with the application of a construction heuristic for obtaining an initial feasible OVRP solution. This solution is then improved by means

of the proposed metaheuristic approach denoted by Broad Local Search Algorithm (BLSA).

3.5.1. Obtaining the initial OVRP solution

Our algorithmic development was applied to the OVRP considering both the hierarchical objective of primarily minimizing vehicles and secondarily the total route length, and the single objective of minimizing the total route length. Depending on the objective considered, our metaheuristic employs a different construction methodology: when the hierarchical objective is taken into account, the construction method primarily aims at generating the minimum number of routes, and secondarily at minimizing the total cost of the generated route set. In specific, the lower bound for the required routes is calculated as $K_{min} = \left\lceil (\sum_{i=1...n} q_i)/Q \right\rceil$, and a set *rts* consisting of K_{min} empty routes is generated. With each route $j \in rts$, is associated the corresponding residual capacity. At each iteration, the method calculates the trial residual capacity for inserting every unassigned customer v_i into every possible route $j \in rts$. The customer-route pair yielding the lowest, non-negative trial residual capacity is identified, and the customer is inserted to the route position that minimizes the insertion cost. If, for a customer, no feasible insertion route exists, an additional empty route becomes available and our metaheuristic method is executed primarily aiming at eliminating any unnecessary routes. This is achieved by penalizing the cost of every arc containing the depot. In specific, we set $c'_{0i} = c_{0i} + M$, where M far exceeds the greatest arc cost in graph A. Therefore, the primary goal of the search is to remove any depot-adjacent arcs from the candidate solution, or in other words, to minimize the required vehicle fleet. As previously mentioned the minimum number of vehicles was obtained through $K_{min} = \left\lceil (\sum_{i=1...n} q_i)/Q \right\rceil$. This bound has also been used by several OVRP researchers in the past [2, 4, 15, 23] when only capacity constraints are imposed on the problem model. For all benchmark instances examined, it proved to be a good lower bound, as it matched the actual number of vehicles in the solutions obtained (see Section 4.3). We must note, however, that for problems which involve customers with large q_i values compared to the vehicle capacity, the

aforementioned bound may be unattainable. In these cases, a better bound could be obtained by heuristically solving the one dimensional bin packing problem, setting the size of the bins equal to Q and the item sizes equal to the customer product demands [23].

When the single objective of route cost minimization is considered, customers are sorted in increasing order of the angle that they form with the depot and a randomly taken radius [28]. Then, customers are selected iteratively to be assigned to some route according to the minimum insertion cost criterion. Note that insertions must respect the capacity constraints posed by the problem, and that when a customer is inserted into an empty route, a new empty route becomes available for subsequent customers.

3.5.2. The adopted tabu and penalization strategies

As will be later presented, the BLSA metaheuristic employs the best-admissible scheme for moving to neighboring solutions, or in other words, solution neighborhoods are exhaustively explored and the local search move minimizing the problem objective function is selected to be applied. This deterministic criterion of moving between solutions causes cycling phenomena to occur. To avoid this risk we have adopted the following rather aggressive tabu strategy which apart from avoiding cycling, proved to effectively diversify the overall metaheuristic: When an SMD instance (either VLBE or 2-opt) is applied to the candidate solution, one of its n_1 and n_2 node values is randomly selected to be declared tabu for a horizon of *tab* algorithmic iterations. Declaring a node tabu implies that every tentative local search move represented by an SMD instance with either its n_1 or n_2 value equal to this node is considered tabu, and therefore not allowed to be applied to the candidate solution. Note that if a tentative move improves the best solution obtained through the search process, the tabu policy is overridden.

To further diversify the conducted search, the BLSA improvement metaheuristic employs a penalization scheme based on the proposed SMD representation of moves. In specific, except for the actual cost tag, every VLBE SMD instance also contains a penalized cost label which is calculated as follows: with each problem vertex v_i is associated a counter $count_i$, responsible for keeping track of the number of times that a VLBE SMD instance with n_1 or n_2 equal to v_i has been implemented to the candidate solution by increasing the solution cost. Thus each time, a cost-increasing VLBE SMD

instance is applied to the candidate solution, both its n_1 and n_2 node counters are augmented by one. The penalized cost tag of every VLBE SMD instance with n_1 = A and n_2 = B is set equal to its actual cost tag augmented by $(count_A + count_B) \cdot pen$, where *pen* is a penalization parameter. Note that as per the discussion on the update rules in 3.3, each time a VLBE SMD with n_1 = A and n_2 = B is applied, every SMD instance with either its n_1 or n_2 node value equal to A and B is updated, thus every penalized cost tag is always calculated via the updated $count_A$ and $count_B$ values.

3.5.3. The BLSA improvement metaheuristic

After the initial OVRP solution is obtained, the SMD instances for the VLBE and 2-opt neighborhood structures are constructed. To reduce both the space required for storing the SMD instances, as well as the computational time required for updating their cost tags, we use the following rationale for generating the SMD instances: we solve the OVRP using an OVRP-modified Clarke and Wright [29] heuristic. Let $z_{C\&W}$ and $K_{C\&W}$ denote the cost and the number of routes of the solution obtained. Then a threshold value θ is calculated as $\theta = \beta\, z_{C\&W} / (n + K_{C\&W})$, where β is the sparsification parameter set to 2.5 as proposed by Toth and Vigo [30] for the CVRP. For every vertex pair v_i and v_j, such that $c_{ij} < \theta$ or v_i is the depot node, we generate a collection of $(\mu + 1)^2 - 1$ VLBE SMD instances (corresponding to every possible pair of bone lengths ranging from 0 to μ, with no SMD created for both bone lengths equal to 0), and a single 2-opt SMD instance. Using this filtering scheme for the generation of SMD instances, the overall search process is drastically accelerated without its quality being affected, as the excluded SMD instances represent tentative moves that are highly unlikely to improve the solution quality.

With each of the generated VLBE SMD instances are associated two cost labels, denoted by *cst* and *p_cst*. The former is always equal to the actual cost for implementing the move represented by the SMD instance, whereas the latter is equal to this cost augmented via the penalization strategy presented in 3.5.2, and is used for diversifying the conducted search. With each 2-opt SMD is associated only the actual cost tag *cst*. These aforementioned cost labels are evaluated according to the status of the initial solution. Note that initially, the *p_cst* label is set equal to the *cst* one. Then, the generated

SMDs are inserted into the appropriate Fibonacci Heaps. Specifically, two heaps are created for the VLBE neighborhood structure. The first heap (FH_{VLBE}) is responsible for keeping the VLBE SMD instances sorted according to their non-penalized cost tag (*cst*), whereas the second one (P_FH_{VLBE}) sorts the VLBE SMD instances according to their penalized cost values (*p_cst*). For the 2-opt operator, a single Fibonacci Heap ($FH_{2\text{-}OPT}$) is created for keeping the 2-opt SMD instances sorted according to their non-penalized cost tags.

After the SMD representation and the Fibonacci Heaps have been prepared, the core of the BLSA improvement methodology is iteratively applied. At each iteration, the minimum cost, feasible and non-tabu, VLBE and 2-opt moves are retrieved from FH_{VLBE} and $FH_{2\text{-}OPT}$, respectively. The lower cost of these two tentative moves is selected to be applied to the solution, if it reduces the cost of the current solution. If none of these two moves are improving, a diversification step is applied: a random value λ uniformly distributed within $[1, \mu]$ is generated, and from the P_FH_{VLBE} heap, we identify the lowest *p_cst* SMD which satisfies the following requirements: it is non-tabu, encodes an inter-route move, and the sum of its *n1_len* and *n2_len* is equal to or greater than λ. These requirements are intended to drastically diversify the search process, as they lead to significant solution structure modifications, and therefore drive the algorithm towards unexplored solution trajectories.

The local search move represented by the selected SMD is applied to the current solution, and the cost tags of the affected SMD instances are updated according to the update rules presented in 3.4. Updating the cost of a single SMD instance involves its deletion from the corresponding Fibonacci Heap, the evaluation of its new cost label, and its re-insertion to the Heap. The cost evaluation and insertion operations are performed in constant time, thus the required computational complexity for a single SMD update is determined by the deletion process which requires logarithmic complexity. The BLSA metaheuristic is terminated when a certain time bound has been reached. The pseudocode of the BLSA method is provided in Table 1.

The BLSA was executed for two different objective configurations: a) hierarchical objective (minimize routes then minimize cost), and b) single objective (minimize cost). As previously mentioned, depending on the objective considered, a different construction

method was employed for obtaining the initial solution. Furthermore, when the single objective is considered, the BLSA framework always maintains an empty route through the search process.

Table 1. Pseudocode of the BLSA

```
OVRP Solution BLSA (OVRP Solution S_0, int μ, int minT, int maxT)
    OVRP Solution S, S', S*
    Fibonacci Heap FH_VLBE, P_FH_VLBE, FH_2-OPT
    SMD twoOpt, vlbe, app

    -- initialization phase
    generate the SMD instances for the VLBE and 2-opt neighborhood structures
    calculate the cost tags for the generated SMD instances according to the state of S_0
    insert the VLBE SMD instances into FH_VLBE and P_FH_VLBE
    insert the 2-OPT SMD instances into FH_2-OPT
    S = S_0
    -- improvement phase
    while (termination condition = false)
        --identify local search move
        vlbe = best, feasible, non-tabu VLBE SMD extracted from FH_VLBE
        twoOpt =  best, feasible, non-tabu 2-opt SMD extracted from FH_2-OPT
        if (vlbe_cst < 0 OR twoOpt_cst < 0)
            if (vlbe_cst < vlbe_cst )
                app = vlbe
            else
                app = twoOpt
            end if
        else
            stochastically generate λ in [0, μ]
            app = best, feasible, non-tabu, inter-route VLBE SMD such that n1_len + n2_len ≥ λ
            Node app1 = app_n1, Node app2 = app_n2
            count_app1 = count_app1 + 1, count_app2 = count_app2 + 1
        end if
        stochasticaly generate tab in [mint, maxT]
        declare app1 and app2 tabu for tab iterations of the outer while loop

        -- apply local search move
        apply app to S so that S' is obtained, S = S'
        for every affected SMD instance aff (according to the update rules of Section 3.4)
            remove aff from the corresponding Fibonacci heap(s)
            calculate aff_cst according to the modified solution state
            if (app is a VLBE SMD instance)
                Node n1 = aff_n1, Node n2 = aff_n2
                aff_p_cst = aff_cst + (count_n1 + count_n2) · pen
            end if
            reinsert aff into the corresponding Fibonacci heap(s)
        end for
        if (cost (S) < cost (S*))
                S* = S
        end if
    end while
    return S*
```

4. Computational Results

In this Section we discuss on the BLSA parameter setting, and analytically present the algorithmic performance on a set of well-known OVRP instances. The BLSA method was implemented in Visual C#, and executed on a single core of an Intel T5500 processor (1.66 GHz). All instances and results can be found at http://users.ntua.gr/ezach/

4.1. Benchmark instances

To tune the algorithmic parameters, as well as to assess the performance of the BLSA metaheuristic, we have tested it on a collection of widely studied OVRP benchmark instances. In specific, we have solved seven OVRP instances taken from the CVRP data set of Christofides et al. [31] involving from 50 to 199 customers (C1-C5 and C11-C12), two instances originally introduced for the CVRP by Fisher [32] (F11-F12), and eight large-scale OVRP test problems (O1-O8) which involve from 200 to 480 customers and were introduced by Li et al. [15]. For all seventeen OVRP instances, the cost matrix is obtained by calculating the Euclidean distances of the vertex locations. Regarding the constraints imposed, all seventeen instances consider vehicles to have a maximum carrying capacity. On the contrary, no restriction is imposed on the total length traveled by a route. Note that we did not solve instances C6-C10 of Christofides et al. [31], because these test problems impose maximum route cost constraints on the OVRP model, which are not considered by the BLSA method. Table 2 presents in detail the seventeen OVRP test problems used for testing the proposed metaheuristic. It contains the problem size n, the maximum carrying load of the vehicles Q, and the lower bound for the number of routes required K_{min}, calculated as shown in 3.5.1.

4.2. Parameter Setting

The BLSA algorithmic framework contains three parameters, namely μ, *tab* and *pen*, the values of which have to be decided before the algorithm is executed. To determine the standard parameter setting of the BLSA metaheuristic, we performed extensive tests solving the Christofides et al. [31] and Li et al. [15] OVRP benchmark instances considering the hierarchical objective function of primarily minimizing the vehicle fleet and secondarily optimizing the total travel distance. The first parameter μ is the

maximum length of the bones considered by the VLBE operator. Obviously, the setting of μ depends on how many customers are assigned to each route, which is an instance-specific characteristic. For example, if each route can service up to 5 customers, there is no meaning into setting μ higher than the value of 5. For the class of seven instances, introduced by Christofides et al. [31], each route services approximately 11 customers, on average. For this set of instances, we executed the BLSA method for values taken from {4, 5, 6, 7, 8}. The best solution scores were observed for μ set to 6, 7, and 8. For the standard parameter setting, we used μ = 6, as the best balance between algorithmic effectiveness and efficiency was observed. The same μ value (6) was also considered for the two instances of Fisher [32]. For the large scale OVRP instances of Li et al. [15] about 40 customers are assigned to each route. Thus, we tested the BLSA performance for greater μ values. In specific, BLSA was executed with μ values taken from [6, 12] interval. Higher μ values (10 to 12) did well in terms on the quality of the final solution produced. However, for these values the algorithm was rather slow, as the computational complexity demanded by BLSA exhibits a cubic dependence on the μ parameter. The value of μ was therefore fixed at 8, for which a satisfactory algorithmic performance was achieved.

Table 2. OVRP benchmark instances

Christofides et al. [31] data set				**Fisher [32] data set**				**Li et al. [15] data set**			
Problem Instance	***n***	***Q***	K_{min}	**Problem Instance**	***n***	***Q***	K_{min}	**Problem Instance**	***n***	***Q***	K_{min}
C1	50	160	5	**F11**	71	30000	4	**O1**	200	900	5
C2	75	140	10	**F12**	134	2210	7	**O2**	240	550	9
C3	100	200	8					**O3**	280	900	7
C4	150	200	12					**O4**	320	700	10
C5	199	200	16					**O5**	360	900	8
C11	120	200	7					**O6**	400	900	9
C12	100	200	10					**O7**	440	900	10
								O8	480	1000	10

n: number of customers, ***Q***: vehicle capacity, K_{min}: Lower bound for the routes required

Regarding the second algorithmic parameter, *tab* controls the horizon for which a node (and correspondingly the associated SMD instance population) is declared tabu. The greater the value of *tab* the stronger is the diversification effect on the search process. Preliminary experiments indicated a more robust and effective performance when *tab*

was stochastically set in every BLSA iteration rather than being fixed throughout the search process. Thus, at each iteration, the proposed method stochastically generates μ values uniformly distributed within [*minT*, *maxT*]. Various intervals were used for generating the *tab* values. In specific several *minT* and *maxT* values were tested from the [3, 10], and [5, 20] ranges, respectively. For all three instance sets, *tab* values uniformly taken from the [3, 12] interval resulted in a satisfactory algorithmic behavior, both in terms of solution quality and speed.

The third and last calibrated parameter *pen* determines the penalization term used for augmenting the penalized cost tags of the VLBE instanced. Apparently, the setting of the *pen* parameter depends on both the cost matrix and the solution characteristics of the instance examined. To capture this dependence the penalization term was expressed as $pen = a \cdot \theta$, where θ is the threshold value used for filtering out poor-quality SMD instances, introduced in 3.5.3. We performed several BLSA executions with *a* values taken from the [0.001, 0.01] range. The best algorithmic behavior was recorded when $0.006 \leq a \leq 0.01$, thus the penalization parameter *pen* was set equal to $0.008 \cdot \theta$.

4.3. Computational results on the OVRP benchmark instances

As earlier mentioned, the BLSA framework was applied to the seventeen OVRP test problems for two different objective configurations: a) the hierarchical objective configuration which primarily aims at minimizing the number of routes required and secondarily at minimizing the total cost of the vehicle routes; and b) the single objective configuration which solely calls for the minimization of the produced route set cost.

4.3.1. Results obtained for the hierarchical objective

To test the proposed algorithm's performance considering the hierarchical OVRP objective function, we applied BLSA ten times to each test problem. Note that the initial solution was the same for all ten algorithmic executions, as the relevant construction method behaves deterministically. After performing some preliminary experiments, for the instances of Christofides et al. [31] and Fisher [32], we set the computational time bound to 600 seconds, which proved adequate for ensuring a thorough exploration of the solution space. For the large scale problems of Li et al. [15], solution improvement was

frequently observed after the 600-second time bound; therefore the termination condition of the algorithm was set to the completion of 1800 seconds. The results obtained are summarized in Table 3. In specific, the first set of columns presents the average values over all ten BLSA executions, whereas the second column group refers to the algorithmic run which managed to obtain the highest quality solution. From the results of Table 3, we can see that BLSA managed to consistently generate OVRP solutions minimizing the required fleet of vehicles, for all benchmark problems. In terms of the routing cost, it has shown adequate stability as the gaps between the best and average solution scores obtained over the ten runs are limited from 0.00% to 0.55%, averaging at a satisfactory 0.13%. The least stable performance, which was observed for instance O5, is mainly attributed to the fact that this instance exhibits tightly binding capacity constraints: the eight routes available through the search process offer a maximum carrying load of 7200 units which is equal to the total customer demand. Thus, the local search conducted by BLSA is confined to a rather narrow feasible search space, without the capability of being drastically diversified by employing significant solution modifications. Regarding the average computational times required for obtaining the best solution of each run, they ranged from 28 seconds for the 50-customer instance C1, up to 1590 seconds for the 360-customer problem O5. The aforementioned computational times are satisfactory, considering both the instance scale and the great cardinality of the neighborhood structures explored. Table 3 also presents the solution cost lower bounds obtained in [26] for the Chrostofides et al. [31] and Fisher [32] instances, and the percent gaps between the best BLSA solution and the corresponding lower bound. From the results, we see that BLSA solved four instances to optimality. The maximum deviation was observed for instance C5 (5.0%), whereas the average deviation was limited to 1.3%.

Table 4 compares the best results obtained by BLSA against those obtained by some of the most effective previously published OVRP approaches. These include the algorithms of Pisinger and Ropke [17], Fleszar et al. [23], Li et al. [15], and Repoussis et al. [20], denoted by ALNS, VNS, ORTR, and H-ES, respectively. In addition the BLSA results are compared to the best known solutions scores denoted by *BK*. Regarding the computational times reported in Table 4, for the ALNS and the VNS methods, we provide the average CPU consumption required for a single algorithmic run. For the H-

ES method, as well as for the proposed BLSA approach, the elapsed time when the best solution was firstly encountered through the search process (and not the total time required for an algorithmic run) is provided. Finally, for the ORTR methodology, the authors do not explicitly point out whether the reported CPU times refer to the total running time of a single algorithmic execution, or the time involved when the best solution was obtained.

Table 3. BLSA results for the hierarchical OVRP objective

	AVG			**BST**			**%Gap**	**LB**	**%GapLB**
Instance	**z**	**K**	**t**	**z**	**K**	**t**			
C1	416.06	5.0	28	416.06	5	25	0.00	416.1*	0.0
C2	568.38	10.0	72	567.14	10	68	0.22	559.2	1.4
C3	639.98	8.0	97	639.74	8	103	0.04	639.7*	0.0
C4	733.93	12.0	204	733.13	12	190	0.11	730.2	0.4
C5	895.62	16.0	332	893.39	16	355	0.25	848.5	5.0
C11	682.34	7.0	76	682.12	7	85	0.03	657.1	3.7
C12	534.24	10.0	47	534.24	10	39	0.00	534.2*	0.0
F11	177.00	4.0	132	177.00	4	93	0.00	177.0*	0.0
F12	770.57	7.0	278	769.55	7	301	0.13	762.9	0.9
O1	6018.52	5.0	635	6018.52	5	612	0.00		
O2	4562.88	9.0	832	4557.38	9	774	0.12		
O3	7735.10	7.0	921	7731.00	7	681	0.05		
O4	7264.32	10.0	1009	7253.20	10	957	0.15		
O5	9243.69	8.0	1590	9193.15	8	1491	0.55		
O6	9824.44	9.0	1108	9793.72	9	1070	0.31		
O7	10363.28	10.0	1094	10347.70	10	1257	0.15		
O8	12430.06	10.0	1273	12415.36	10	1512	0.12		
average							*0.13*		*1.3*

AVG: Column set referring to the average values over the ten BLSA executions
BST: Column set referring to the BLSA run which generated the highest quality solution
z: the cost of the generated route set
K: the number of routes present in the solution
t: time elapsed when the best solution was produced
%Gap: the percent gap between the best and average solution cost (= 100·(*AVG-BST*)/*AVG*)
BLSA was coded in Visual C# and executed on a single core of a T5500 processor (1.66 GHz)
LB: the cost lower bound reported in [26] (instances solved to optimality are marked with *)
%GapLB: the percent gap between the best BLSA cost and the LB values (= 100·(*BST-LB*)/*BST*)

As seen from Table 4, BLSA was successful to improve eight of the seventeen previously best-known solutions. For the other nine test problems, it consistently produced solutions matching the best-known ones. The average solution improvement is equal to 0.06% (0.01% for the small-scale instances, and 0.11% for the large-scale ones).

In particular, for the large scale instances of Li et al. [15], BLSA robustly improved seven out of the eight instances, while for the smallest instance O1, it matched the previously best reported solution score. In terms of the computational times required by the algorithms compared, we do not intend to perform an analytic comparison, as the running times depend on a variety of computational factors which are difficult to be securely compared. Furthermore, for the presented previous approaches, no detailed information is always provided by the authors on whether the presented CPU times refer to the time required for the algorithms to run to completion, or to the CPU time elapsed when the best solutions were obtained. However, as an overall comment, we observe that the ORTR and VNS methods appear to be faster than the other three approaches.

Table 4. Comparison of effective metaheuristics for the hierarchical OVRP objective

	BK	**ALNS**		**VNS**		**ORTR**		**H-ES**		**BLSA**		**%Gap**
Instance		**z**	**t**			**z**	**t**	**z**	**t**	**z**	**t**	
C1	416.06(5)	**416.06**	23	**416.06**	1	**416.06**	6	**416.06**	98	**416.06**	25	0.00
C2	567.14(10)	**567.14**	53	**567.14**	1	**567.14**	31	**567.14**	143	**567.14**	68	0.00
C3	639.74(8)	641.76	128	**639.74**	12	**639.74**	40	**639.74**	330	**639.74**	103	0.00
C4	733.13(12)	**733.13**	279	**733.13**	29	**733.13**	129	**733.13**	613	**733.13**	190	0.00
C5	894.11(16)	896.08	237	905.96	15	924.96	381	894.11	1272	***893.39***	355	0.08
C11	682.12(7)	**682.12**	141	**682.12**	12	682.54	122	**682.12**	318	**682.12**	85	0.00
C12	534.24(10)	**534.24**	118	**534.24**	8	**534.24**	33	**534.24**	363	**534.24**	39	0.00
F11	177.00(4)	**177.00**	104	178.09	7	**177.00**	20	**177.00**	264	**177.00**	93	0.00
F12	769.55(7)	770.17	359	769.66	62	769.66	158	**769.55**	753	**769.55**	301	0.00
O1	6018.52(5)	-		-		**6018.52**	365	**6018.52**	452	**6018.52**	612	0.00
O2	4583.70(9)	-		-		4584.55	440	4583.70	613	***4557.38***	774	0.57
O3	7731.46(7)*	-		-		7732.85	493	7733.77	736	***7731.00***	681	0.01
O4	7260.60(10)*	-		-		7291.89	574	7271.24	833	***7253.20***	957	0.10
O5	9197.61(8)	-		-		9197.61	767	9254.15	1365	***9193.15***	1491	0.05
O6	9803.80(9)	-		-		9803.80	977	9821.09	1213	***9793.72***	1070	0.10
O7	10348.57(10)*	-		-		10374.97	935	10363.40	1547	***10347.70***	1257	0.01
O8	12420.16(10)*	-		-		12429.56	1127	12428.20	1653	***12415.36***	1512	0.04
average												*0.06*

BK: Objective value of the best known solution, followed by the required number of vehicles in the parentheses
ALNS: Adaptive Large Neighborhood Search of Pisinger and Ropke [17] - Pentium IV 3GHz
VNS: The Variable Neighborhood Search of Fleszar et al. [23] - Pentium M 2 GHz
ORTR: The Record to Record Algorithm of Li et al. [15] - Athlon 1GHz
H-ES: The Hybrid Evolutionary Algorithm of Repoussis et al. [20] - C++, Pentium IV 2.8GHz
BLSA: The proposed Broad Local Search Algorithm - Visual C#, T5500 1,66 GHz
z: the cost of the generated route set
t: CPU times reported for the algorithmic executions
%Gap: the percent gap between the best BLSA solutions and the previously best-known solution scores (= 100·(*BK*-*BLSA*)/*BK*)
* Solution scores obtained by the ABHC algorithm of Derigs and Reuter [18] for 400000 iterations.
Bold characters represent higher quality solutions, whereas bold and italic characters represent new best-known solutions obtained by the proposed BLSA method.

4.3.2. Results obtained for the single objective

The algorithm was executed ten times on each test problem considering the single objective. For the single objective, each algorithmic run involved a different initial solution, as the construction procedure works stochastically (customers are arbitrarily

sorted for being assigned to the routes). The computational time bounds were again fixed at 600 seconds for the instances of Christofides et al. [31] and Fisher [32], and 1800 seconds for the large scale problems of Li et al. [15]. Table 5 presents the results obtained. The first column set presents the average values over all ten BLSA executions, whereas the second group of columns refers to the BLSA run which yielded the best solution score. Again, BLSA appears to be adequately stable, as the percent deviation between the best and average solution scores over the ten runs ranged from 0.00% to 0.31%, averaging at 0.14%. Regarding the average CPU times required for obtaining the best solution of each algorithmic execution, they varied from 30 seconds for the 50-customer problem C1, up to 1414 seconds for the 480-customer instance O8.

Table 5. BLSA results for the single OVRP objective

	AVG			**BST**			**%Gap**
Instance	**z**	**K**	**t**	**z**	**K**	**t**	
C1	412.96	6.0	30	412.96	6	24	0.00
C2	564.06	11.0	62	564.06	11	55	0.00
C3	641.20	8.8	98	639.26	9	106	0.30
C4	734.92	12.0	179	733.13	**12**	167	0.24
C5	871.67	17.0	268	869.00	17	256	0.31
C11	679.27	9.3	103	678.54	9	87	0.11
C12	534.24	10.0	32	534.24	**10**	29	0.00
F11	177.00	4.0	103	177.00	**4**	83	0.00
F12	763.10	8.0	229	761.68	8	189	0.19
O1	5996.71	6.0	583	5988.35	6	648	0.14
O2	4560.32	9.0	770	4549.46	**9**	804	0.24
O3	7733.65	7.0	843	7731.00	**7**	864	0.03
O4	7258.07	10.0	997	7251.30	**10**	1058	0.09
O5	9176.85	9.0	1107	9152.47	9	956	0.27
O6	9819.56	9.0	1220	9793.72	**9**	1180	0.26
O7	10362.44	10.0	1206	10347.70	**10**	1332	0.14
O8	12428.91	10.0	1414	12412.26	**10**	1582	0.13
average							*0.14*

AVG: Column set referring to the average values over the ten BLSA executions
BST: Column set referring to the BLSA run which generated the highest quality solution
z: the cost of the generated route set
K: the number of routes present in the solution
t: time elapsed when the best solution was produced
%Gap: the percent gap between the best and average solution cost (= $100 \cdot (AVG\text{-}BLSA)/AVG$)
BLSA was coded in Visual C# and executed on a single core of a T5500 processor (1.66 GHz)
Bold characters used in the K column of the BST group denote solutions that require $\boldsymbol{K_{min}}$ vehicles

Table 6 compares the best results obtained by BLSA for the single objective configuration against the results obtained by previously published OVRP solution methodologies which also considered the aforementioned objective. These are the BATA [12], LBTA [13] and the BoneRoute [14] approaches. Note that comparisons are made

only for the instances of Christofides et al. [31], as to our knowledge, this is the first time that the single objective is considered for the Fisher [32], and the Li et al. [15] large-scale OVRP instances. The BLSA method improved the previously best reported solutions for three out of the seven examined instances. For the other four test problems, it robustly matched the best solution scores previously published. The average improvement over the previously published best solutions is equal to 0.02 %, which is satisfactory considering the small size of the Christofides et al. [31] instances, and the effectiveness of the LBTA, BATA and BoneRoute methods.

Table 6. Comparison of effective metaheuristics for the single OVRP objective

	BAS	**BATA**		**LBTA**		**BR**		**BLSA**		**%Gap**
Instance		**z**	**t**	**z**	**t**	**z**	**t**	**z**	**t**	
C1	412.96(5)	**412.96**	6	**412.96**	23	**412.96**	98	**412.96**	24	0.00
C2	564.06(10)	**564.06**	31	**564.06**	53	**564.06**	143	**564.06**	55	0.00
C3	639.57(8)	642.42	40	639.57	128	641.77	330	***639.26***	106	0.05
C4	733.68(12)	736.89	129	733.68	279	735.47	613	***733.13***	167	0.07
C5	869.24(17) *	879.37	381	870.26	237	877.13	1272	***869.00***	256	0.03
C11	678.54(10)	679.60	122	**678.54**	141	679.38	318	**678.54**	87	0.00
C12	534.24(10)	**534.24**	33	**534.24**	118	**534.24**	363	**534.24**	29	0.00
average										*0.02*

BAS: Score of the best algorithmic solution previously reported, followed by the number of solution routes in the parentheses
BATA: The Backtracking Adaptive Threshold Accepting Algorithm of Tarantilis et al. [12] - Pentium II 400MHz
LBTA: The List Based Threshold Accepting method of Tarantilis et al. [13] - Pentium II 400MHz
BR: The BoneRoute algorithm of Tarantilis et al. [14] - Pentium II 400MHz
BLSA: The proposed Broad Local Search Algorithm - Visual C#, T5500 1,66 GHz
z: the cost of the generated route set
t: CPU times reported for the algorithmic executions
%Gap: percent gap between the scores of the BLSA solutions and the previously reported best solutions (=100·(*BAS*-*BLSA*)/*BAS*)
***** These solution scores were obtained by the ABHC algorithm of Derigs and Reuter [18] for 400,000 iterations.
Bold characters represent higher quality solutions, whereas bold and italic characters represent new best-known solutions obtained by the proposed BLSA method.

5. Conclusions

This article presented an original OVRP method which examines rich neighborhood structures formed by exchanging large customer sequences, instead of single customer vertices. This is achieved at the expense of manageable computational effort, as the proposed metaheuristic adopts the static move descriptor (SMD) concept for reducing the complexity of local search based-methods. The central idea of the SMD concept is that when moving from one solution to another, only a limited solution part is modified; therefore to examine the subsequent neighborhoods, only the tentative moves that refer to this modified solution part must be re-evaluated. On the contrary, moves associated with

the unaffected solution characteristics do not need to be re-evaluated, if appropriately stored. The conducted local search is diversified via a tabu strategy and a penalization scheme both of them compatible with the static representation of tentative moves.

The proposed algorithm (abbreviated by BLSA) was applied to a collection of seventeen well-known OVRP benchmark instances considering two different objective configurations. The first one primarily aims at minimizing the vehicle fleet required, and secondarily at minimizing the routing cost, whereas the second objective only aims at the minimization of the cost of the routes produced. For both objective configurations, BLSA managed to improve several previously reported best-known solutions.

In terms of future research directions, the static move descriptor concept can be applied to numerous combinatorial optimization applications. Furthermore, effective and time-consuming neighborhood structures could be encoded into SMD instances to be incorporated into efficient local-search frameworks. Finally, the SMD design can be extended to contain feasibility information, so that infeasible tunneling is allowed through the search process.

Acknowledgments

We are indebted to the anonymous referees, for extensively reviewing our paper, and offering constructive remarks and directions for the completion of our work.

References

[1] Li F, Golden B, Wasil E. Very large-scale vehicle routing: new test problems, algorithms, and results. Comput Oper Res 2005;32(5):1165-79.

[2] Fu Z, Eglese RW, Li LYO. A new tabu search heuristic for the open vehicle routing problem. J Oper Res Soc 2005;56:267–74.

[3] Fu Z, Eglese RW ,Li LYO. Corrigendum. A new tabu search heuristic for the open vehicle routing problem. J Oper Res Soc 2006;57:1018.

[4] Brandão J. A tabu search algorithm for the open vehicle routing problem. Eur J Oper Res 2004; 157(3): 552-64.

[5] Russell R, Chiang W-C, Zepeda D. Integrating multi-product production and distribution in newspaper logistics. Comput Oper Res;35(5):1576-1588.

[6] Bodin L, Golden B, Assad A, Ball M. Routing and scheduling of vehicles and crews : The state of the art. Comput Oper Res 1983;10(2):63-211.

[7] Fu Z, Wright M. Train plan model for British Rail freight services through the Channel Tunnel. J Oper Res Soc 1994;45:384–91.

[8] Li LYO, Fu Z. The school bus routing problem: a case study. J Oper Res Soc 2002;53:552–58.

[9] Zachariadis EE, Kiranoudis CT. A Strategy for Reducing the Computational Complexity of Local Search-Based Methods, and its Application to the Vehicle Routing Problem. Under Review. Available at: http://users.ntua.gr/ezach/

[10] Schrage L. Formulation and structure of more complex/realistic routing and scheduling problems. Networks 1981;11(2):229-32.

[11] Sariklis D, Powell S. A heuristic method for the open vehicle routing problem. J Oper Res Soc 2000;51(5):564-573.

[12] Tarantilis CD, Ioannou G, Kiranoudis CT, Prastacos GP. A threshold accepting approach to the open vehicle routing problem. RAIRO 2004;38:345–60.

[13] Tarantilis CD, Kiranoudis CT, Ioannou G, Prastacos GP. Solving the open vehicle routing problem via a single parameter metaheuristic algorithm. J Oper Res Soc 2005;56:588–96.

[14] Tarantilis CD, Diakoulaki D, Kiranoudis CT. Combination of geographical information system and effective routing algorithms for real life distribution operations. Eur J Oper Res 2004;152:437–53.

[15] Li F, Golden B, Wasil E. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. Comput Oper Res 2004;34(10):2918-30.

[16] Dueck G. New optimization heuristics: the great deluge algorithm and the record-to-record travel. J Comput Phys 1993;104:86–92.

[17] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. Comput Oper Res 2007;34(8):2403-35.

[18] Derigs U, Reuter K. A simple and efficient tabu search heuristic for solving the open vehicle routing problem. J Oper Res Soc; doi:10.1057/jors.2008.107.

[19] Whittley IM, Smith GD. The Attribute Based Hill Climber. J Math Model Algor 2004;3(2):167-78.

[20] Repoussis PP, Tarantilis CD, Bräysy O, Ioannou G. A hybrid evolution strategy for the open vehicle routing problem. Comput Oper Res;doi:10.1016/j.cor.2008.11.003.

[21] Glover F. Future paths for integer programming and links to artificial intelligence. Comput Oper Res 1998;13:533–49.

[22] Voudouris C, Tsang E. Guided local search. Eur J Oper Res 1998;113:80–119.

[23] Fleszar K, Osman IH, Hindi KS. A variable neighbourhood search algorithm for the open vehicle routing problem. Eur J Oper Res 2009;195:803-9.

[24] Hansen P, Mladenović N. Variable neighborhood search: Principles and applications. Eur J Oper Res 2001;130:449-67.

[25] Li X-Y, Tian P, Leung SCH. An ant colony optimization metaheuristic hybridized with tabu search for open vehicle routing problems. J Oper Res 2008; In press (doi:10.1057/palgrave.jors.2602644).

[26] Letchford AN, Lysgaard J, Eglese RW. A branch-and-cut algorithm for the capacitated open vehicle routing problem. J Oper Res Soc 2007;58:1642-51.

[27] Fredman M, Tarjan R. Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM 1987; 34:596-615.

[28] Cordeau J-F, Gendreau M, Laporte G. A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems. Networks 1997;30:105-119.

[29] Clarke G, Wright JW. Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 1964; 12: 568–89.

[30] Toth P, Vigo D. The granular tabu search (and its application to the vehicle routing problem). INFORMS J Comput 2003;15:333-48.

[31] Christofides N, Mingozzi A, Toth P. The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P, Sandi C, editors. Combinatorial optimization. Chichester, UK: Wiley; 1979. p. 315–38.

[32] Fisher M. Optimal solution of vehicle routing problems using minimum k-trees, Oper Res 1994;42:626-42.